CITRIX®

Xen and Client Virtualization: the case of XenClient XT

Gianluca Guida
<gianluca.guida@citrix.com>

Overview

- A Xen perspective on Client Virtualization
- XenClient XT architecture
- XenClient XT and open source
- XenClient XT as a “less insecure” platform

Client Virtualization

What is it?

It has become a successful marketing buzzword

- Used in very different contexts, meaning pretty much everything
- Confuses engineers
- Restricting definition in this talk

Client Virtualization

What do you mean by it?

«An hypervisor as transparent as possible running in end-user devices. »

Why?

- Increased security by isolating in different VMs public, personal and confidential data
- Virtual Machines are easier to backup, restore, migrate

Client Virtualization

Why it is different from Server Virtualization

- **No client/server architecture**

  Hypervisor runs in the same machine where the user interacts with guest VMs.
- **One-to-few as opposed to many-to-many users/VMs**

  While multiple users are supported, only one user at the time can interact with her/his guest VMs
- **No standardized Virtual Machine model**
  - The user expects OEM features and machine-specific hardware support like non-standard embedded devices (and e-mail buttons!)
  - Machine-specific graphics, 3D and sound are a very important part of today's interfaces
- **Physically interacting with the machine influences VMs**

  (e.g., USB plug/unplug, low battery notification, external display attach)

Xen and Client Virtualization

Is Xen a good architecture for Client Virtualization?

- **Architecture and design of Xen is a big advantage**
  - Modularity makes it incredibly adaptable
  - Provides a solid base to achieve security and isolation
- **Xen is a very active open source project!**
  - Modern hardware support (TPM, Power and Thermal Management, IOMMU, trusted boot)
  - Upstream Linux kernel support
    - Privileged VM kernels have support for many End-User Devices (3G modems, bluetooth, fingerprint readers)
  - Upstream QEMU support
    - State-of-the-art device emulation, easily extensible and modifiable

Meet the XenClient Family

- **Experimental project at Citrix (2008)**
  - Safe pass-through of graphics, extensive use of IOMMU
  - Exposing and virtualizing modern laptops` hardware in VMs
- **XenClient 1.0 (2010)**

  “Virtual Desktops To-go”: Client virtualization on mobile computers and infrastructure for central management and deployment of VMs.
- **XenClient XT released with XenClient 2.0 (2011)**

  XT is the hardened version of XenClient, using hardware, cryptography and architectural changes to maximize security
- **Citrix acquires VirtualComputer: XenClient Enterprise (2012)**

  Product lines in process of merging. XCE gives wider hardware support, XC provides innovative graphics and GPU virtualization techniques.

  XT is the most extensible version of the XenClient product line.

XenClient XT

Main Features and Design Goals

- **Client virtualization with focus on security goals**
- **3D GPU access to one VM**
- **Enforce isolation among VMs and devices**
- **Protect user data stored physically in the machine**
- **Platform to support third-party services to VMs**

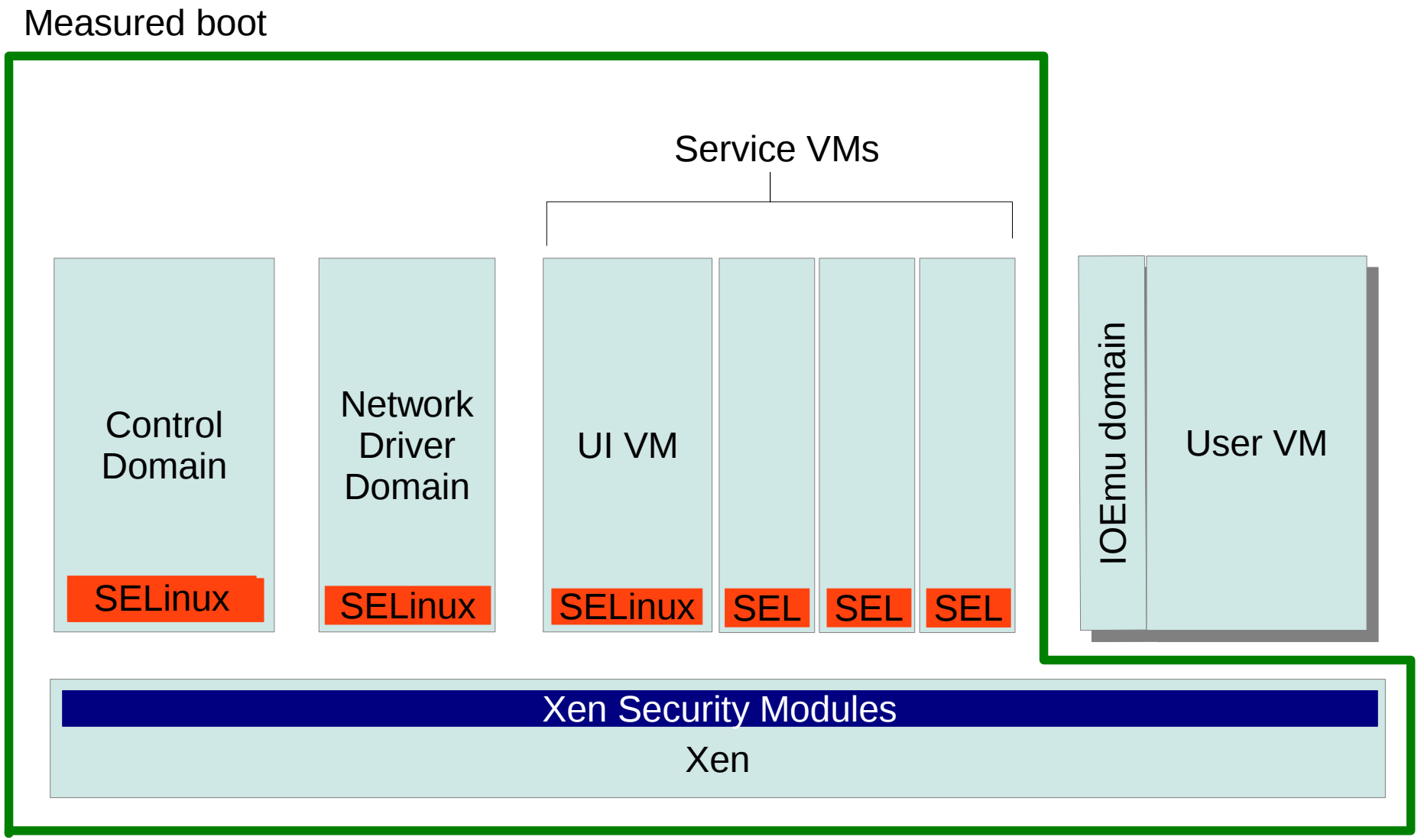
XenClient XT

Main Features and Design Goals (cont'd)

Not covered here:

- Ability to safely and quickly backup over the network VMs to a Data Center
- Data Center tools to manage, deploy and update VM images in distributed laptops

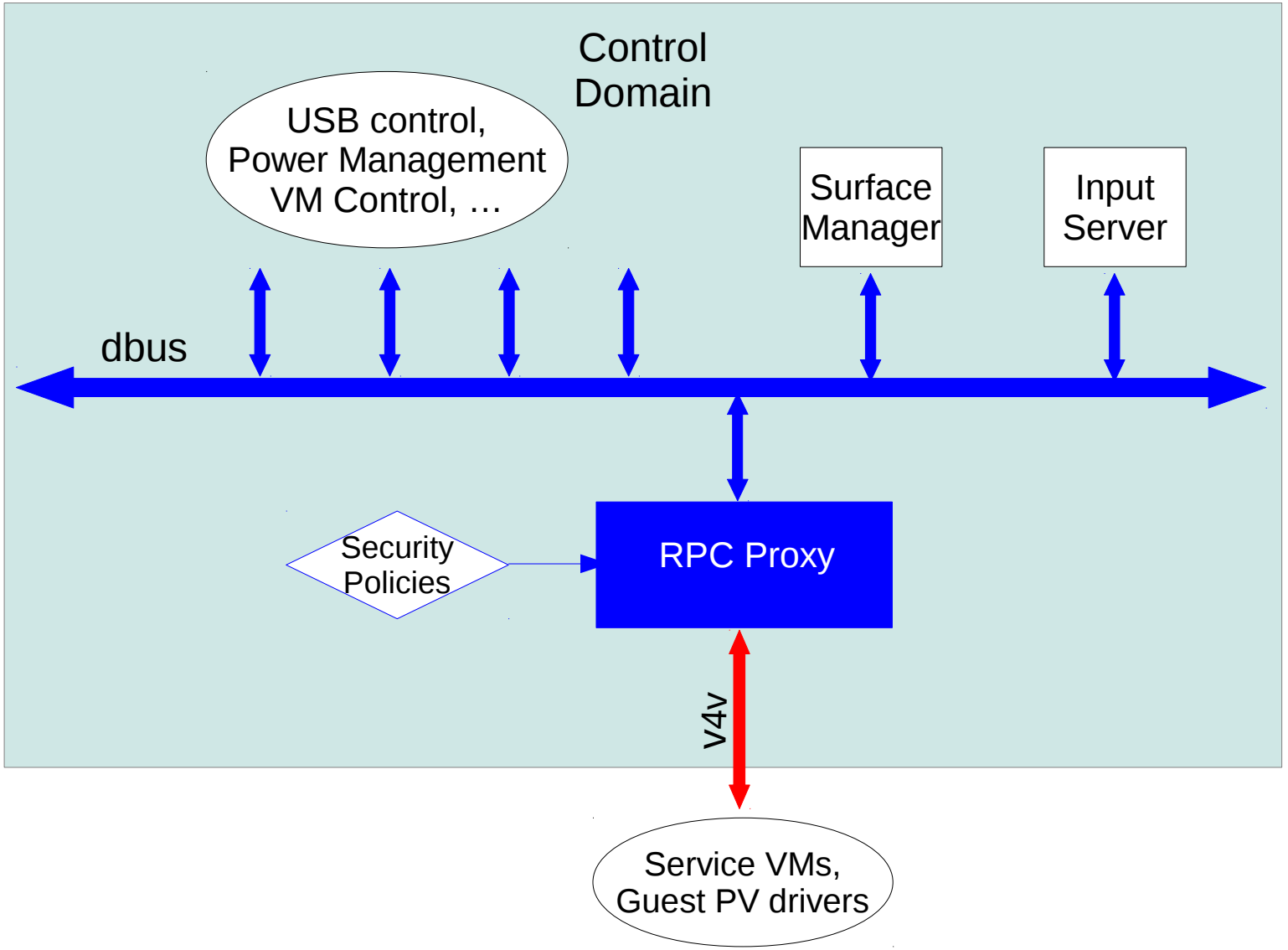
XenClient XT Architecture



XenClient XT Architecture

- **Stub Domains for I/O Emulation**
  - On Linux not minios
- **Network drivers in separated Driver Domain**
  - Multiple Network Driver Domains possible
- **Extensive use of SELinux and XSM policies**
  - Enforced in production
- **DRTM Xen launch**
  - Measurement of root filesystems of Service and Control VMs and grub2 command line
- **User Interface separated from Control Domain**
  - User doesn't access for normal operations a privileged VM.

Control Domain



Control Domain

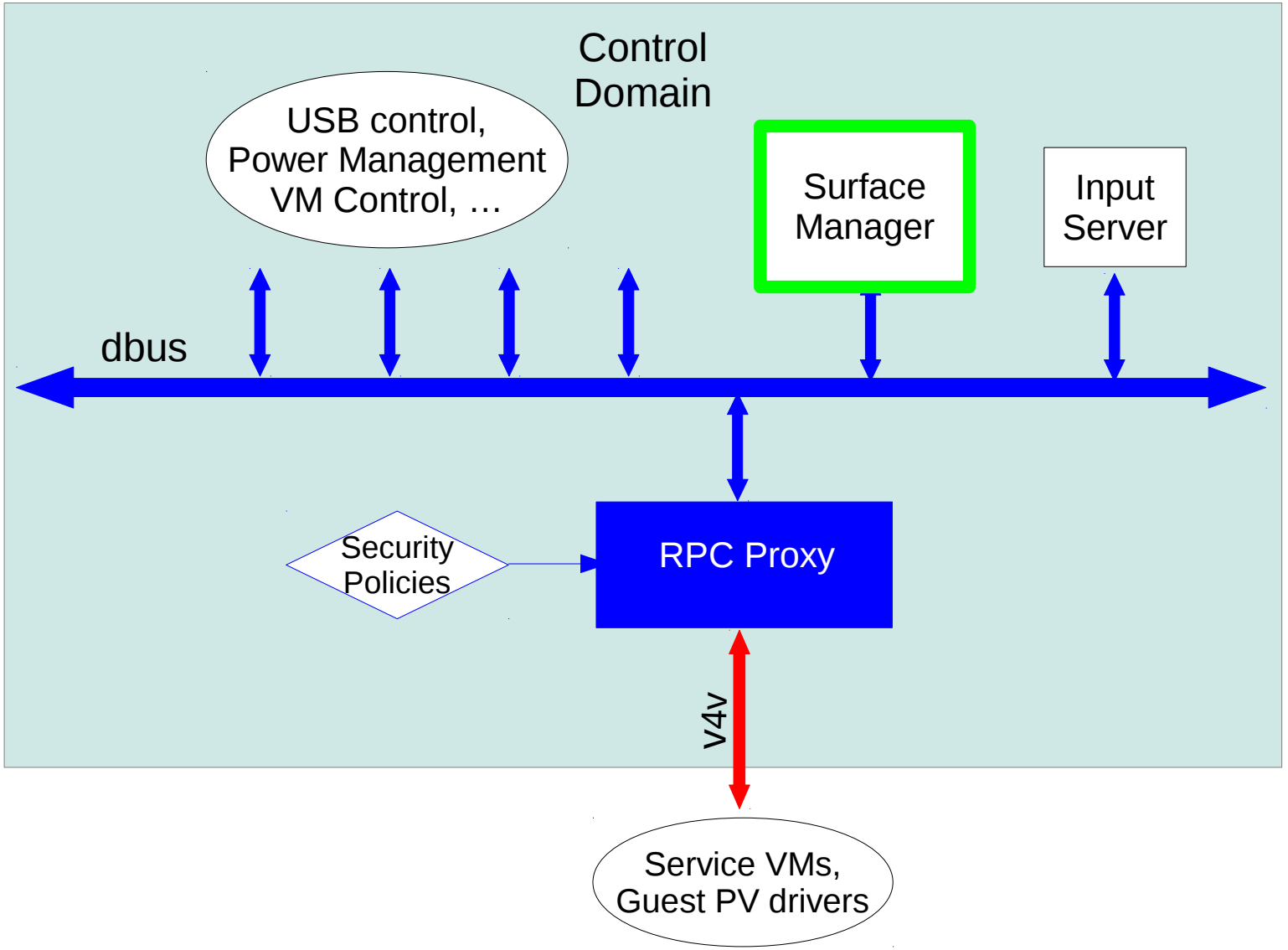
- **Component based architecture**
  - dbus RPC
- **XenClient Toolstack**
  - Per-VM and global daemons, high level language
- **Proxy to inter-domain bus (v4v)**
  - Allows transparent disaggregation
  - Security policies control external domains interaction
- **Daemons multiplex machine resource to VMs**
  - Input Server: keyboard, mouse, user interaction
  - Surfman: display and graphics
  - Other things not in talk (USB manager and per VM daemon, ...)

v4v as inter-domain bus

- **Copy-based inter-domain communication**

- **(domain, port, protocol) namespace**
  - Port defines the service, protocol (DGRAM or STREAM) the message type
    - Subtle difference with TCP/IP namespace
- **One-to-one and one-to-many support**
  - DOMID_ANY, server can accept multiple clients on a defined port
- **Linux socket-based interface**
  - Minimal interaction in the kernel, mostly implemented in userspace library, as direct into Xen as possible, to ensure isolation.
- **Interposition library**
  - LD_PRELOAD and ssh → sshv4v
- **Works on Windows, too**
  - Guest tools can communicate with Control Domains and other services.

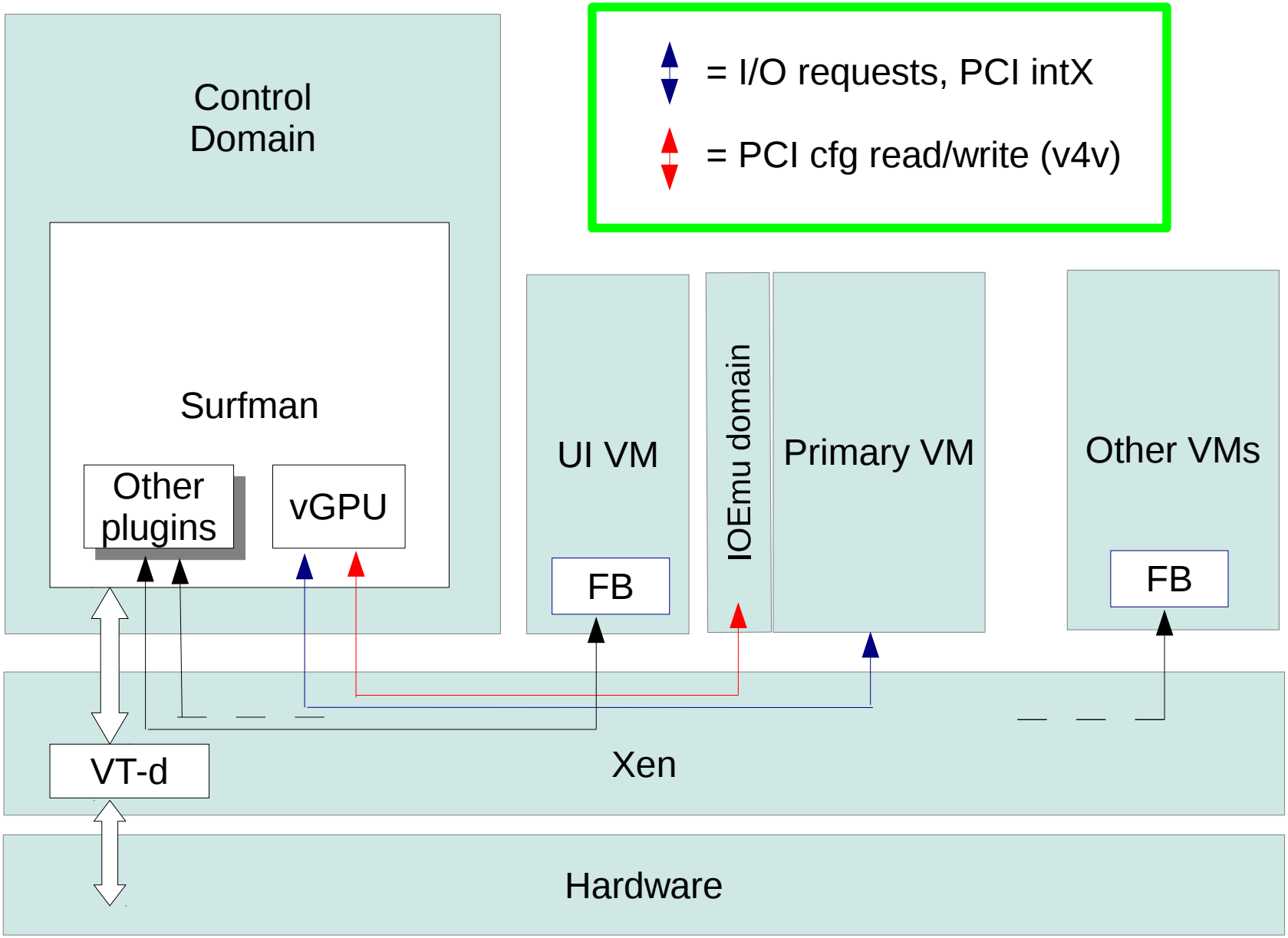
Surface Manager



Surface Manager

- Controls the laptop screen and external display
- Plugin architecture
  - Each plugin has a different way of virtualizing graphics
  - Hardware dependent plugins can be added on specific machines
  - Stable API to support binary plugins
- Control virtual machines' frame buffers
  - Switch between VMs

Surface Manager



QEMU disaggregation

- **vGPU Surfman plugin**
  - Non-QEMU PCI virtual device
- **Multiple IO Requests servers support in Xen**
- **Needs hook in QEMU to forward PCI cfg access**
  - Unnecessary dependancy
  - v4v as a device model bus complicates design

QEMU disaggregation

Decouple PCI configuration and BDF enumeration from QEMU

- Xen aware of PCI I/O space
- Virtual PCI device registers static BDFs in Xen
- Xen dispatches PCI config access to the correct IOREQ

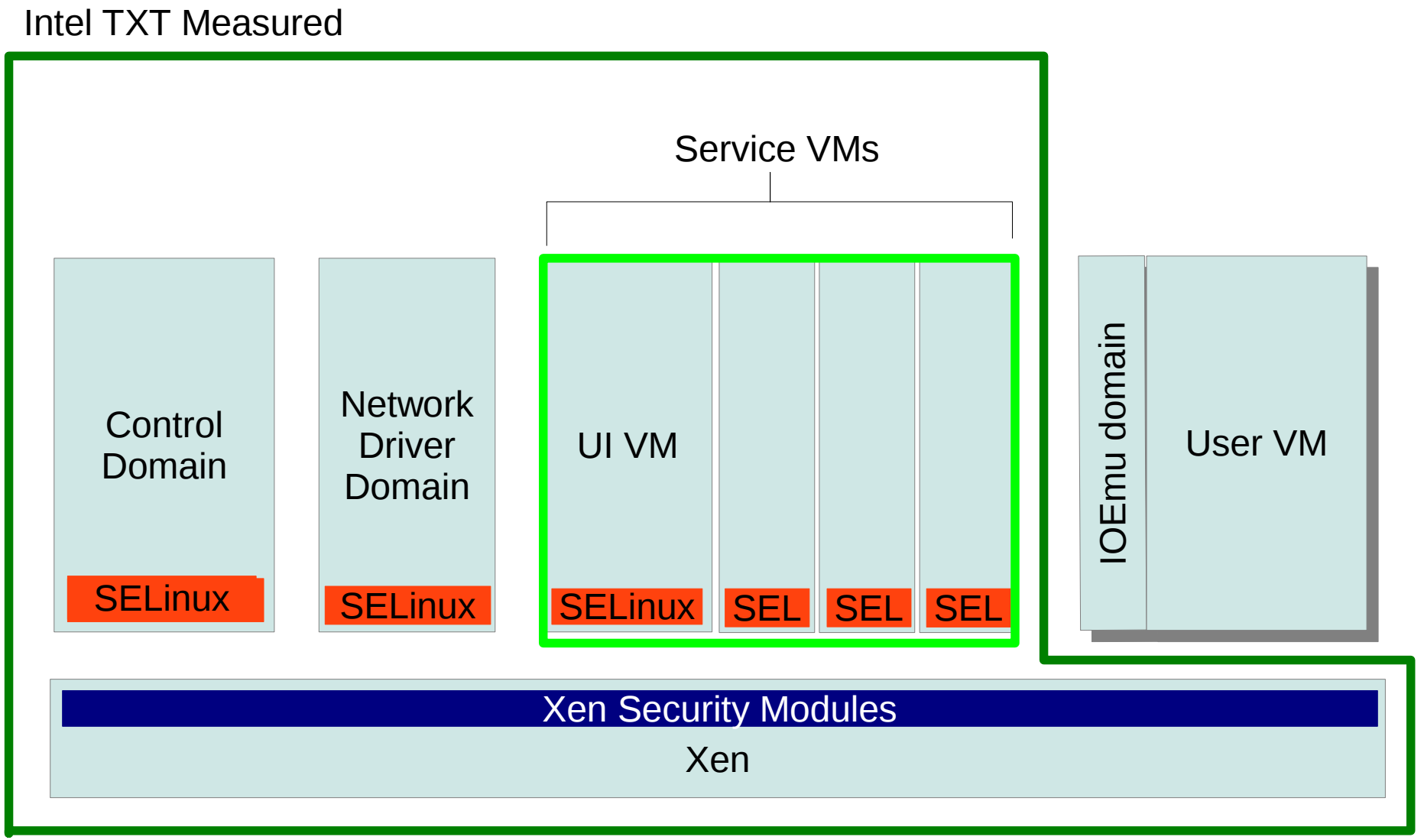
Why we should do that

- Quickly prototyping virtual devices
- Add platform-specific devices without hacking/forking QEMU upstream
- Disaggregate more complex code

Granularity of disaggregation limited

- Some devices are coupled together
- Disaggregation most probably done in a per-class basis

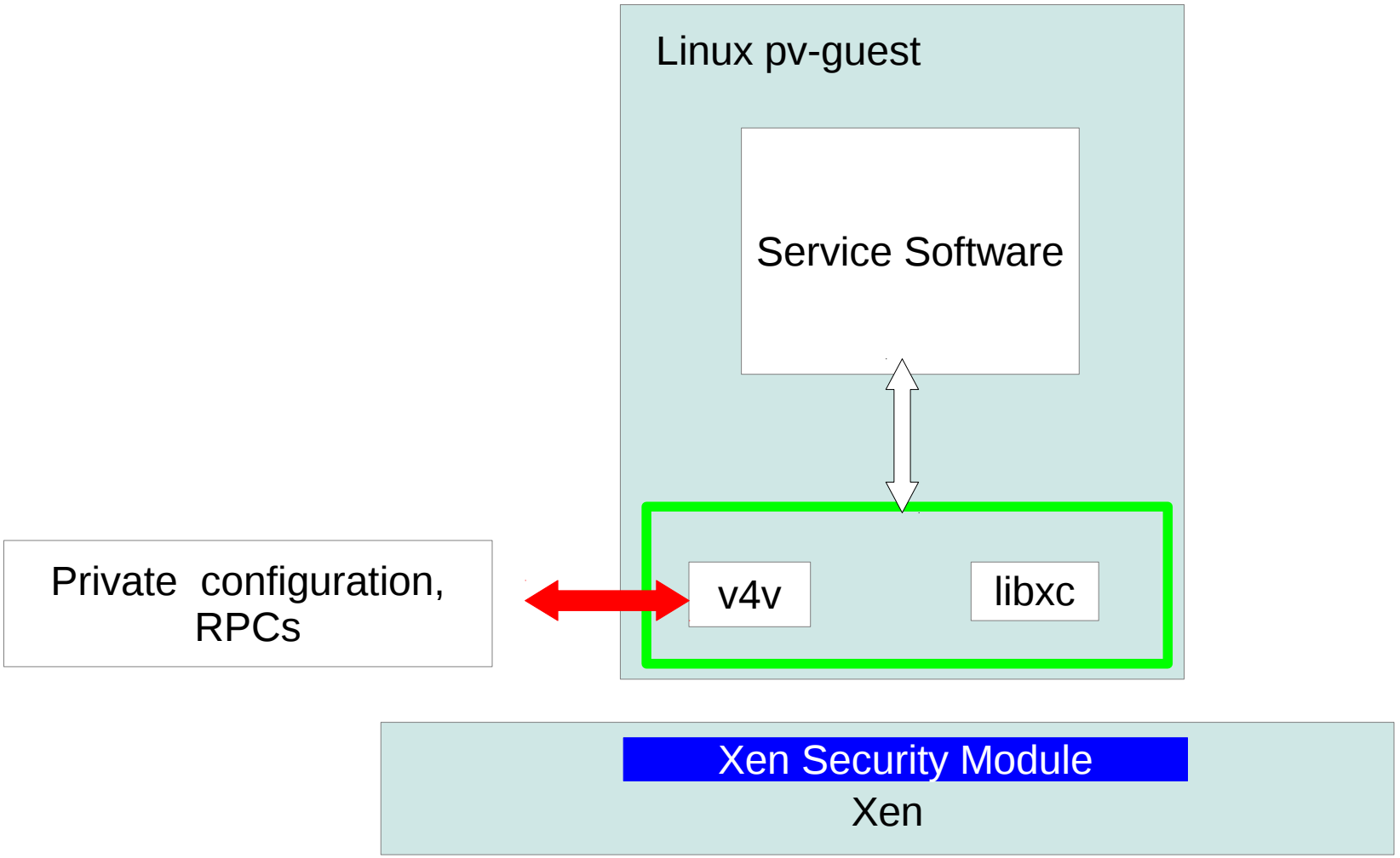
XenClient XT as a “less insecure” platform



XenClient XT as a “less insecure” platform

- **Control RPC and v4v proxy**
  - Separated domains can interact with the system and provide services
- **XSM and RPC security policies**
  - Control domain can limit in a fine-grained way interaction of service VMs with the system
- **Boot-time measurements of Service VMs**
  - Avoid giving special permissions to compromised VMs

Anatomy of a Service VM



Anatomy of a Service VM

- Linux PV-guest with Service software
- Service API Library
  - Access to private configuration database in Control Domain
  - Interaction with RPC system
  - Issue permitted Xen hypercalls

VPN VM

Service VM example

- netback
  - Toolstack connects guest pv-interface to it
- Vpn software
  - Certificates and configuration might be stored in Control Domain private configuration
- Netfront
  - Encrypted data sent to Network Driver Domain

VPN VM

- **Guest is unaware of the VPN**
- **A compromised OS in guest do not compromise VPN access keys**
- **Unencrypted data only from guest to VPNVM**
  - The Network Driver Domain network stack sees only encrypted data
  - Network Driver Domain becomes security non-critical
    - It is the most exposed component to threats!
- **Network path VM ↔ NDVM is stackable**
  - e.g. IDS VM, nested VPN (?)

Third party Service VMs

- Service VMs are not only network related
- Security policies can give to service VMs a lot of flexibility
  - Even selected privileged hypercalls in selected domains
- Service VMs should not be XenClient XT only!
  - It is definitely possible to create an API and a format to describe Service VMs in an hypervisor-agnostic, portable way